PROBABILISTIC GRAPHICAL MODEL FOR FLASH MEMORY PROGRAMMING

Borja Peleato, Rajiv Agarwal and John Cioffi

Electrical Engineering Department
Stanford University, CA 94305-9510

ABSTRACT

Flash memory presents significant advantages over hard drives in terms of read speed and power efficiency; however its lifetime can be several orders of magnitude smaller. Thus increasing lifetime of flash memory via signal processing techniques is an important research area. The first half of the paper presents a statistical method for estimating the health of the cells in a Flash memory, based on which a variable error correction coding scheme can be used to increase lifetime. The second half of the paper proposes a statistical approach to increase lifetime when the flash controller can dynamically vary the program and erase operation strategy. This approach uses Markov Decision Processes (MDP) to choose the optimal program or erase strategy at any given point in the life of a Flash memory based on its current state or health. From a bigger picture stand-point, the paper presents a novel way of flash management using a Markov model for health of the memory at any given point in its lifetime.

***Index Terms*—** Flash memories, Markov decision process, hidden Markov model, probabilistic graphical model.

1. INTRODUCTION

Recently, flash memories have emerged as a faster and more efficient alternative to hard drives.

NAND-flash memories have unique characteristics that pose challenges to the SSD system design, especially the aspects of random write performance and write endurance. They are organized in terms of blocks, each block consisting of a fixed number of pages. A block is the elementary unit for erase operations, whereas reads and writes are processed in terms of pages [1]. Before data can be written to a page, the block must have been erased. Moreover, NAND flash memories can undergo a limited program-erase (PE) cycle count, or equivalently there is a limit to the number of times information can be re-written. Typically, flash chips based on single-level cells (SLC) sustain 10^5 and those based on multi-level cells (MLC) 10^4 PE cycles.

Each cell in a page is a floating gate transistor which can be charged to a specific voltage value by injecting voltage pulses into the floating gate. Similarly, blocks are erased by injecting pulses with negative voltage [2]. The programming and erasing pulses damage the dielectric barrier of the cells, increasing its permeability. The increased permeability causes leakage, which ultimately limits the endurance of the memory. The amount of damage that a cell suffers depends on the amount of charge that is pushed through its dielectric barrier. Cells programmed to higher voltages therefore suffer more damage than those programmed to the lower ones. However, the dielectric barrier can partially repair itself during rest periods [3]. Damaged cells are also more susceptible to voltage pulses, requiring fewer pulses to charge but with a higher risk of over-programming.

Thus there are several factors that control and affect the cells charge-carrying capacity or its health. This paper presents a Hidden Markov Model (HMM) to capture these factors and estimate the health of a cell in a tractable manner. The proposed model is used in two different ways to increase the lifetime of the memory. In the first half, a state estimation algorithm is proposed to estimate current health based on previous health and easily observable parameters. This knowledge of current health can be used in several ways to increase Flash lifetime. In the second half of the paper, a Markov Decision Process (MDP) model is used to optimize the program and erase strategy for every PE cycle.

2. SYSTEM MODEL

A directed probabilistic graphical model consists of a collection of nodes representing variables and edges representing dependencies between the variables. An edge going from node a to node b represents a direct dependency of the variable associated with node b on the variable associated with node a. If the only path between two sets of nodes S and T passes through a third set of nodes U, the variables associated to nodes in S are independent of those associated to nodes in T given those associated to nodes in U. Several more complicated dependencies can be captured by the graphical representation, and can be found in [4].

For ease of exposition, the models in this paper will represent write and erase operations on a single cell. Since all the cells in a page are programmed and erase simultaneously, the same model can be used for all of them. First, a model will be built to represent a single erase and write operation on a cell. Then, several instances of this model will be concatenated to form a second model representing multiple PE cycles.

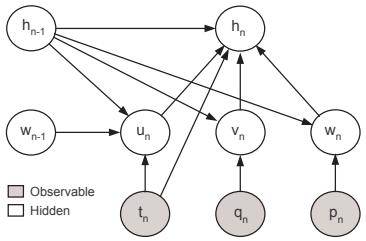

Fig. 1. Graphical model representing the variables involved in erasing and re-writing a cell in a flash memory.

The directed graphical model proposed to represent the nth PE cycle of a cell is shown in Figure 1. The node labeled w_{n-1} represents the voltage to which the cell was programmed during the previous cycle, and t_n represents the time elapsed between the end of that cycle and the beginning of the nth cycle. During this time t_n, some charges have leaked, resulting in a voltage u_n when a new write request comes. The cell then needs to be erased and reprogrammed. First, the cell is erased using pulses of predetermined amplitude q_n, resulting in a voltage v_n. Then the cell is reprogrammed to a voltage w_n using pulses of predetermined amplitude p_n. The erase, write and leakage processes are noisy, so u_n, v_n and w_n are random variables.

The programming and erase procedure damages the cell. The node labeled h_n represents the health of the cell after the nth cycle, and the damage suffered is captured as a decrease in value from h_{n-1} to h_n. The amount of damage does not depend on the amplitude of the pulses directly but on the amount of charge that is pushed through the dielectric barrier, i.e. $u_n - v_n$ and $w_n - v_n$. However, larger pulses have a higher chance of over-programming, hence causing more damage. The model captures this indirect effect as a path from (p_n, q_n) to h_n, through (v_n, w_n). The edge from t_n to h_n represents the healing during the rest period t_n.

Knowing the health of a cell, or rather, of the cells in a page, can be of great use in practice. However, the health of a cell cannot be measured. It is a hidden variable used to label the state of a cell into a discrete set of predefined values that go from being fresh (maximum health) to dead (minimum health). Similarly, this paper will assume that p_n and q_n can only take a pre-defined set of values. For illustrative purposes, the paper will refer to these values as pulse amplitudes, but they could also be labels for different pulse shapes, number of pulses, etc. In practice, the voltages w_{n-1}, u_n, v_n and w_n can be read, albeit only in discrete steps by comparing them to reference voltages. Reading them, however, is impractical, hence they are treated as hidden variables.

The goal of the later sections will be to study how these variables evolve with the program and erase operations of the memory. The exact dependencies between the variables are different depending on the cell size, flash manufacturer, etc. The first step would therefore be to decide the number of values used to quantize each variable, based on the designers experience, and take measurements to characterize the device.

Unfortunately, the joint distribution for the hidden variables is too complex to be estimated directly. For example, assuming a 3-bit quantization for each variable, the table for $f(h_n, u_n, v_n, w_n | h_{n-1}, w_{n-1}, t_n, q_n, p_n)$ would have over 10^8 entries. The graphical model in Figure 1 solves this by expressing it as a product of easily estimated factors:

- $\phi_u^n = f(u_n | h_{n-1}, w_{n-1}, t_n)$: decrease in cell voltage as a function of time and cell health (leakage).
- $\phi_v^n = f(v_n | h_{n-1}, q_n)$: erased cell voltage as a function of erase pulses and cell health (erase).
- $\phi_w^n = f(w_n | h_{n-1}, p_n)$: programmed cell voltage as a function of program pulses and cell health (program).
- $\phi_h^n = f(h_n | h_{n-1}, u_n, v_n, w_n, t_n)$: health of the cell as a function of the previous health, the initial, erased and programmed voltages and the time elapsed since the last programming (damage and healing).

Under a 3-bit quantization for each variable, the number of entries to be estimated in these factors is in the order of 10^5, which is much more manageable. Furthermore, since each factor represents a well known process, they can sometimes be modeled independently using only a few parameters.

The graphical model in Figure 1 represents the nth erase and reprogram operation on a cell. The same model can be repeatedly concatenated to represent the evolution of the cell through its lifetime. Let $s_n = (h_n, u_n, v_n, w_n)$ and $r_n = (t_n, q_n, p_n)$ denote the state of the cell and the observed variables at the nth PE cycle, respectively.

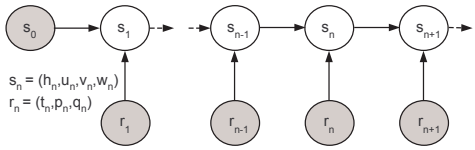

Fig. 2. Graphical model representing multiple PE cycles.

The graphical model in Figure 2 represents the temporal evolution of a page. The initial state s_0 consists of the values that (h, u, v, w, t) had before the cell was first programmed, which are known. The subsequent states are unknown, but their distribution can be inferred from the observations r_n and the state transition probability $f(s_n | s_{n-1}, r_n)$.

3. STATE ESTIMATION

This section will focus on estimating the pdf (probability density function) of the state of a page after n PE cycles, for a given sequence of observations $(r_1, r_2, \ldots, r_n)$ and a given state transition probability $f(s_n | s_{n-1}, r_n)$. The pdf of s_n can be useful, for example, to estimate the probability of error, leakage rate and remaining lifetime. Those estimates can in turn be used to adapt parameters such as the code rate, the

voltage levels, background re-writes frequency, etc. Some wear leveling and bad block management techniques could also take advantage of knowing the health of a given block or page.

The probability distribution function of s_n for a given sequence of observations can be decomposed according to the graphical model in Figure 2 as:

$$\beta_s^n = \sum_{s_{n-1}} f(s_n|s_{n-1}, r_n)\beta_s^{n-1}, \quad (1)$$

where $\beta_s^n = f(s_n|r_1, \ldots, r_n)$. If the initial state s_0 is known, it is possible to iterate Eq. 1 in a forward recursion [5] to find $f(s_n|r_1, \ldots, r_n)$ for all n. However, the state transition probability $f(s_n|s_{n-1}, r_n)$ is hard to characterize directly. The graphical model provides a way of overcoming this obstacle: the state transition probability can be decomposed as $f(s_n|s_{n-1}, r_n) = \phi_u^n \phi_v^n \phi_w^n \phi_h^n$.

This factorization of the state transition probability makes it possible to simplify Eq. 1 as follows.

$$\beta_s^n = \sum_{h_{n-1}, w_{n-1}} \phi_u^n \phi_v^n \phi_w^n \phi_h^n \alpha_{(h,w)}^{n-1},$$

where $\alpha_{(h,w)}^{n-1} = f(h_{n-1}, w_{n-1}|r_1, \ldots, r_{n-1})$. Similarly,

$$\alpha_{(h,w)}^n = \sum_{h_{n-1}, w_{n-1}, u_n, v_n} \phi_u^n \phi_v^n \phi_w^n \phi_h^n \alpha_{(h,w)}^{n-1}$$

for $n \geq 1$ and $\alpha_{(h,w)}^0$ is known from the initial state.

4. PULSE OPTIMIZATION

If the controller can choose the amplitude of the programming and erase pulses at each PE cycle, the graphical model can be used to find a sequence of pulses that increase the memory speed, lifetime and reliability. Strong pulses provide faster program and erase operations but smaller ones cause less damage and are more reliable. However, as the health of the cell decreases, it becomes more susceptible to voltage pulses, i.e. the same pulse would cause greater damage and voltage increase to a damaged cell than to a healthier one.

Let $R(s, a)$ represent the expected reward obtained when the controller takes an action a on a page in state s. $R(s, a)$ could include factors such as the cell damage, probability of error or delay. This section proposes a method for finding the sequence of actions a_n, $n = 1, 2, \ldots$ that maximize the expected sum of rewards obtained in the process illustrated in Figure 2, under the dynamics found in section 3. This problem can be posed as a Markov Decision Process (MDP) and solved using value or policy iteration [5]. In an MDP, the action taken at the nth step depends only on the state at that time, not on any previous history, hence the name. The mapping from states to actions is known as a policy. For each policy $\pi : s \to a$, it is possible to define a value function over the states through the recursive set of equations

$$V^\pi(s_n) = R(s_n, \pi(s_n)) + \gamma E[V^\pi(s_{n+1})], \quad (2)$$

where the expectation is taken over s_{n+1} and γ is a discount factor between 0 and 1 that favors immediate rewards over long term ones. For each state s, $V^\pi(s)$ approximates the expected sum of rewards obtained by acting according to π when s is the initial state.

There is a one-to-one correspondence between policies and value functions. Given a value function $V(s)$, its associated policy is

$$\pi^V(s_n) = \arg\max_{a_{n+1}} E[V(s_{n+1})]. \quad (3)$$

The optimal policy and value functions, i.e. the ones which maximize $E\left[\sum_{n=1}^{\infty} R(s_{n-1}, a_n)\right]$, can be found using the value iteration algorithm [5], which consists on initializing $V(s) = 0$ for all s and iterating

$$V(s_n) := \max_a R(s_n, a) + \gamma E[V(s_{n+1})] \quad (4)$$

for all possible values of s_n until convergence. The optimal policy is then found using Eq. 3.

In the example presented here, the action are the pulse amplitudes $a_n = (p_n, q_n)$ and the state is re-defined as $s_{n-1} = (h_{n-1}, w_{n-1}, t_n)$. The difference between this state definition and the one given in section 2 lies in the voltages u_{n-1}, v_{n-1} and the rest period t_n. The rest period t_n can be observed but cannot be chosen by the designer. The voltages (u_{n-1}, v_{n-1}) have not been included in s_{n-1} because they cannot play a significant role in the choice of a_n, since (p_n, q_n) are shared by all the cells in a page while (u_{n-1}, v_{n-1}) are different for each cell. Eq. 4 then becomes

$$V(s_{n-1}) := \max_{p_n, q_n} \Big[R(s_{n-1}, p_n, q_n) + \gamma \sum_{s_n} f(s_n|s_{n-1}, p_n, q_n) V(s_n) \Big], \quad (5)$$

where $f(s_n|s_{n-1}, p_n, q_n) = f(t_{n+1}) \phi_w^n \sum_{u_n, v_n} \phi_h^n \phi_u^n \phi_v^n$. The next section presents simulation results for this method.

5. EXPERIMENTS

This section presents an example which attempts to achieve a trade-off between maximizing lifetime and write speed using the MDP approach proposed in section 4. It is expected that the state transition probability will vary from one vendor of flash to another. Hence, the simulations are performed for representative $f(s_n|s_{n-1}, p_n, q_n)$, instead of using distributions applicable for a particular vendor's flash only. Pages start with health $h_0 = 100$ and are considered dead when their health falls below $h_{\text{dead}} = 10$. Each PE cycle reduces

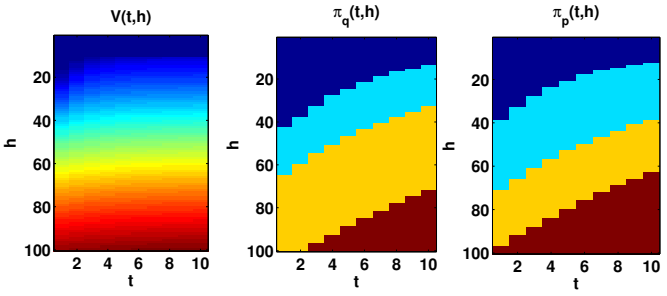

Fig. 3. Optimal value function (left) and policy (center and right) associated with an exponential decrease in health.

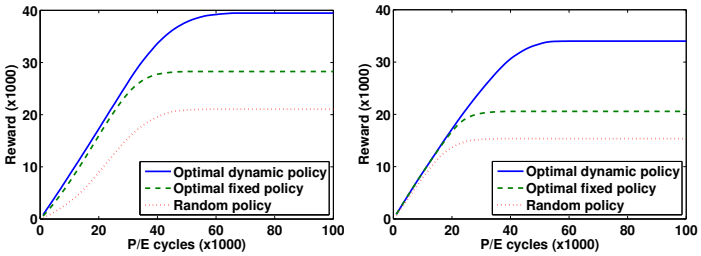

Fig. 4. Cumulative sum of rewards for exponential (a) and quadratic (b) cell damage model.

the health parameter by a random amount. This amount was assumed to be exponentially distributed, with mean that decreases with t and increases with h, w, p, q. The time between consecutive writes t was assumed to be uniformly distributed between 1 and 10 seconds. The pulse amplitudes for p and q were chosen from predetermined sets of four values.

Let $L(h) = 1$ for $h \geq h_{\text{dead}}$ and $L(h) = 0$ otherwise, and let $D(h, p, q) = c\frac{h^4}{pq}$ be a penalty between 0 and 1 associated to the delay introduced by reprogramming a page with health h using program and erase pulses of amplitude p and q, respectively. Then, the desired trade-off between lifetime and write speed can be found using the value iteration algorithm described in Eq. 5 with the reward function $R(h, a) = L(h)(1 - D(h, p, q))$.

The resulting value function and optimal policy are shown in Figure 3. The value function and optimal pulse amplitude increase with h and t, as expected. The policies for p and q both show four clearly separated regions, corresponding to the four different values that p and q can take.

Figure 4a shows the cumulative sum of rewards obtained during 100k PE cycles, averaged over 100 simulation instances. As a framework for comparison, the figure also shows the cumulative sum of rewards for two other policies: in the optimal fixed policy, the pulse amplitudes are optimized to be kept constant for all PE cycles. In the random policy, the pulse amplitudes for each PE cycle are chosen at random from the set of possible values. The slope of the curve at a given PE cycle indicates the expected reward obtained at that cycle. Since the reward decreases with the write delay, steeper curves represent faster writing and the curve becomes flat once all 100 instances have died. The proposed method provides faster writing and increases lifetime by 50% and 40% as compared to the optimal fixed policy and the random policy, respectively.

Finally, to illustrate the wide applicability of the proposed method, it is used for a different damage model, where the probability of a decrease in health falls quadratically instead of exponentially. Figure 4(b) shows the results. For the quadratic model, increase in lifetime is over 50% with respect to both the optimal fixed and the random policy if the proposed MDP method is used.

6. CONCLUSION

This paper proposes a probabilistic graphical model, specifically a hidden Markov model for the program and erase operation of flash memories. The graphical model provides a simple way of characterizing the dependencies between variables, which can the be used to solve a diverse set of problems. This model was used to formulate the choice of program and erase pulses as a Markov decision process, which was then optimized using the value iteration algorithm. Simulations have shown that such an approach can provide significant increases in lifetime and speed over a scheme using fixed pulse amplitudes. The proposed model can be extended in several ways to capture more factors that flash behavior. One such direction currently being studied by the authors is to include inter-cell-interference (ICI) in the model by extending the graphical model to include several adjacent pages. As the NAND flash geometry continues to shrink (state-of-the-art is 19nm), NAND flash lifetime will be severely affected by factors such as write noise from program pulses and ICI, and the proposed graphical model can be a very useful tool.

7. REFERENCES

[1] R. Agarwal and M. Marrow, "A closed-form expression for write amplification in nand flash," in *GLOBECOM Workshops (GC Wkshps)*. IEEE, 2010, pp. 1846–1850.

[2] B. Peleato and R. Agarwal, "Maximizing mlc nand lifetime and reliability in the presence of write noise," in *Proc. of IEEE International Conf. Comm.*, 2012.

[3] R. Yamada, T. Sekiguchi, Y. Okuyama, J. Yugami, and H. Kume, "A novel analysis method of threshold voltage shift due to detrap in a multi-level flash memory," in *VLSI Technology, Symposium on*. IEEE, 2001, pp. 115–116.

[4] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*, The MIT Press, 2009.

[5] D.P. Bertsekas, *Dynamic programming and stochastic control*, vol. 125, Academic Pr, 1976.